On Armstrong-compliant Logical Query Languages

Marie Agier, Christine Froidevaux, Jean-Marc Petit, Yoan Renaud, Jef Wijsen

On Armstrong-compliant Logical Query Languages

Marie Agier
LIMOS UMR 6158 CNRS
Université Blaise Pascal, France
agier@isima.fr

Christine Froidevaux
Université Paris-Sud, CNRS
LRI, UMR 8623, France
chris@lri.fr

Jean-Marc Petit
Université de Lyon, CNRS
LIRIS, UMR 5205, INSA Lyon, France
jean-marc.petit@insa-lyon.fr

Yoan Renaud
Université de Lyon, CNRS
LIRIS, UMR 5205, INSA Lyon, France
yoan.renaud@insa-lyon.fr

Jef Wijsen
Université de Mons, Institut d'Informatique
Mons, Belgium
Jef.Wijsen@umons.ac.be

ABSTRACT

We present a simple logical query language called $\mathcal{RL}$ for expressing different kinds of rules and we study how this language behaves with respect to the well-known Armstrong's axioms. We point out some negative results, e.g. it is undecidable to know whether or not a query from this language is "Armstrong compliant". The main contribution of this paper is to exhibit a restricted form of $\mathcal{RL}$-queries – yet with a good expressive power – for which Armstrong's axioms are sound. From this result, this sublanguage turns out to have structural and computational properties which have been shown to be very useful in data mining, databases and formal concept analysis.

1. INTRODUCTION

The notion of *rules* or *implications* is very popular and appears in different flavors in databases, data mining or artificial intelligence communities. Famous examples of rules are functional dependencies [4], implications [9] or association rules [3]. As such, a simple remark can be made on such rules: their syntax is the same but their semantics widely differs.

In this paper, we consider rules to be defined on *tabular datasets*. Basically, a tabular dataset is equivalent to a set of *relations* over a set $\mathcal{U}$ of distinguished attributes (or columns) in databases terminology [1].

Logical languages can be used to express the different well-known rule semantics on tabular datasets. By the way, a natural and "generic" definition of a query language devoted to "rules" can be elaborated in order to be able not only to capture most of existing semantics already known on tabular datasets (e.g. functional dependencies), but also to devise new semantics specific to some application domains through the key notion of *queries*.

We have also chosen to focus on a class of queries that satisfy Armstrong's axioms, so-called Armstrong compliance. More precisely, Armstrong compliance of a query Q means that on every input relation r, if $ans(Q, r)$ – the answer of Q against r – includes a set of rules F and the rule $X \rightarrow Y$ can be inferred from F by Armstrong's axioms, then $ans(Q, r)$ also contains $X \rightarrow Y$.

For functional dependencies and implications, Armstrong's axioms are known to be sound and complete but more surprisingly, many other semantics also fit into this framework [2]. Roughly speaking, our aim is to define syntactical boundaries of Armstrong-compliant semantics expressed by queries.

Practical interests are for instance that some forms of *reasoning* can be performed on rules (e.g. implication problem in linear time [5]). Moreover, it is also possible to work on "small" *covers* of rules [11, 12, 16] and to use data mining techniques specific to the considered cover, but applicable to *every* Armstrong-compliant queries.

Paper contribution. We extend the relational domain calculus with **attribute-variables** and **schema-variables** that range over attributes and sets of attributes respectively. A simple logical language called $\mathcal{RL}$ for expressing different kinds of rules is presented. This ongoing work brings a logical view to the contribution of [2]. In this paper, a logical query language is proposed from which new undecidability results are proposed. Its main contribution is to exhibit a restricted form of $\mathcal{RL}$-queries – yet with a good expressive power – for which Armstrong's axioms are sound.

From this result, this sublanguage enjoys structural and computational properties which has been shown to be very useful in data mining, databases and formal concept analysis.

LID 2011 March 25, 2011, Uppsala, Sweden

Related works. Declarative query languages for data mining have been studied for years [13, 15, 18, 19]. Logical query languages for data mining have been studied for example in [6, 10]. In [6] , a general query language for data mining has been devised in a different context: the authors defined a data mining language with schema-variables that range over sets of n-ary tuples of attributes. Their objective was to characterize data mining queries amenable to a levelwise search strategy, i.e. exhibiting monotone properties with respect to some partial order. They obtain negative results pointing out that their class of queries was too expressive to ensure properties such as (anti-)monotonic property.

Other declarative approaches have been proposed in data mining but in a much more general setting, i.e. at the intersection of DBMS and data mining techniques (classification, clustering, pattern mining ...), e.g. [8, 18, 19].

An inductive logic programming query language for database mining was also proposed in [20]. The proposal for an inductive logic programming query language puts inductive logic programming into a new perspective. Recent works bridging the gap between constraint programming and data mining have been proposed, for example see [21].

From an application point of view related to gene expression data in biology, an ad-hoc rule language has been proposed in [2] to deal with the high number of semantics biologists could define over their gene expression data to specify their own rules. The problem was not tackled with a query language target as we propose in this paper.

Paper organization. In the following section,we present syntax and semantics of the logical query language $\mathcal{RL}$. Section 3 introduces what is an Armstrong-compliant query language. In Section 4, we show some negative results on this language, mainly the undecidability of Armstrong-compliance of $\mathcal{RL}$-queries. We have to restrict the shape of $\mathcal{RL}$-language to succeed in an Armstrong-compliant one. We describe in Section 5 how to define this new language (called $\mathcal{RLR}$) by considering step by step counter-examples of the original one. Then we show that $\mathcal{RLR}$-queries are Armstrong-compliant. This last result establishes the fact that $\mathcal{RLR}$-queries are amenable to levelwise strategy to compute answer of such queries. Finally, Section 6 concludes this paper.

2. SYNTAX AND SEMANTICS OF $\mathcal{RL}$-LANGUAGE

The $\mathcal{RL}$-language is defined as an extension of domain relational calculus in databases [14] with addition of tuple-variables and schema-variables.

2.1 Alphabet and formulas

For simplicity, we first assume that the database consists of a single relation, avoiding to introduce predicate symbols.

Let $\mathcal{U}$ be a set of attributes. A schema R is a finite, nonempty set of attributes from $\mathcal{U}$. A tuple $\overline{t}$ over a schema R is a total function from R to $\mathcal{CST}$, $\overline{t}[\overline{A}]$ denotes the value of $\overline{t}$ for attribute $\overline{A}$. A relation over a schema R is a finite set of tuples over R.

Then, let us define some notations for the $\mathcal{RL}$-language:

- $\mathcal{CST}$ is a set of constants,
- $s, t, u, s_1, \ldots$ are tuple-variables,
- $A, B, C, A_1, B_1 \ldots$ are attribute-variables, i.e. capital letters from the beginning of the alphabet,
- $X, Y, Z, X_1, Y_1 \ldots$ are schema-variables, i.e. capital letters from the end of the alphabet

To avoid ambiguity with variables, we shall use the following notations for attributes, set of attributes and tuples:

- $\overline{A}, \overline{B}, \overline{C}, \overline{A_1}, \overline{B_1} \ldots$ are single attributes,
- $\overline{X}, \overline{Y}, \overline{Z}, \overline{X_1}, \overline{Y_1} \ldots$ are set of attributes.
- $\overline{s}, \overline{t}, \overline{t_1}, \overline{t_2}, \ldots$ are tuples,

Atomic $\mathcal{RL}$ formulas. Let A, B be attribute-variables, t, s tuple-variables, a a constant, X a schema-variable.

Definition 1. The following expressions are atomic $\mathcal{RL}$ formulas:

$$A = B, \quad t.A = s.B, \quad t.A = a, \quad X(A), \quad A = \bar{A}$$

$\mathcal{RL}$-formulas are defined inductively as follows.

Definition 2.

1. Every atomic $\mathcal{RL}$-formula is a $\mathcal{RL}$-formula
2. If δ_1 and δ_2 are $\mathcal{RL}$-formulas, then $\neg\delta_1$ and $(\delta_1 \wedge \delta_2)$ are $\mathcal{RL}$-formulas
3. If δ is a $\mathcal{RL}$-formula and A an attribute-variable, then $\forall A(\delta)$ is a $\mathcal{RL}$-formula
4. If δ is a $\mathcal{RL}$-formula and t a tuple-variable, then $\forall t(\delta)$ is a $\mathcal{RL}$-formula
5. If δ is a $\mathcal{RL}$-formula, then (δ) is a $\mathcal{RL}$-formula

Other logical connectors such as $\vee, \Rightarrow$, quantifier $\exists$ and abbreviations **true, false** are defined as usual.
Note that schema variables cannot be quantified.
A $\mathcal{RL}$-formula is *closed* if all occurrences of tuple- and attribute-variables are *bound*. To make expressions of $\mathcal{RL}$-formulas easier, we shall use the following abbreviations [6]:

- $\forall t_1, \ldots, t_n(\delta)$ for $\forall t_1(\ldots(\forall t_n(\delta)\ldots)$
- $\forall A(X)(\delta)$ for $\forall A(X(A) \Rightarrow \delta)$
- $\exists A(X)(\delta)$ for $\exists A(X(A) \wedge \delta)$

2.2 The $\mathcal{RL}$-query language

Now we introduce the notion of $\mathcal{RL}$-queries which allows us to express different kinds of rules.

Definition 3. The $\mathcal{RL}$-query language is defined as the set of $\mathcal{RL}$-queries Q_δ expressed as follows:

$$Q_\delta = \{\langle X, Y\rangle \mid \underbrace{\forall \vec{t}(\psi(X, Y, \vec{t}) \wedge (\delta_1(X, \vec{t}) \Rightarrow \delta_2(Y, \vec{t})))}_{\delta(X,Y)}\}$$

where:

- X, Y are free schema-variables of $\delta(X, Y)$.
- $\forall \vec{t}$ is a vector of universally quantified tuple-variables, let us say $\vec{t} = \langle t_1, \ldots, t_n\rangle$
- $\psi(X, Y, \vec{t})$ is a $\mathcal{RL}$-formula with $X, Y, t_1, \ldots, t_n$ free variables,
- $\delta_1(X, \vec{t})$ and $\delta_2(Y, \vec{t})$ are two $\mathcal{RL}$-formulas with respectively $X, t_1, \ldots, t_n$ and $Y, t_1, \ldots, t_n$ free variables .

By definition of $\mathcal{RL}$-queries, we cannot have schema-variables in formulas, except those used for representing the left- and right-hand sides of the rules. A $\mathcal{RL}$-query is denoted Q_δ or simply Q when δ is clear from the context.

Example 1. Let us consider functional dependencies (FD). They can be defined as follows:

$Q_{f_1} = \{\langle X, Y\rangle \mid \forall t_1, t_2(\forall A(X)(t_1.A = t_2.A) \Rightarrow \forall B(Y)(t_1.B = t_2.B))\}$

$Q_{f_2} = \{\langle X, Y\rangle \mid \forall t_1, t_2(\exists A(Y)(\neg X(A)) \wedge \forall A(X)(t_1.A = t_2.A) \Rightarrow \forall B(Y)(t_1.B = t_2.B))\}$

Trivial FD are allowed with Q_{f_1} and disallowed with Q_{f_2}.

Example 2. Let us consider implications as defined in formal concept analysis (domain of attributes should be $\{0, 1\}$): $Q_2 = \{\langle X, Y\rangle \mid \forall t(\forall A(X)(t.A = 1) \Rightarrow \forall B(Y)(t.B = 1))\}$

Let us now consider the following query Q_3 more specific than Q_2 and the relation r:

r	g_1	g_2	g_3	g_4	*Gender*
$\overline{t_1}$	1	0	1	1	M
$\overline{t_2}$	0	1	1	1	F
$\overline{t_3}$	0	1	0	1	F
$\overline{t_4}$	1	1	1	0	M

$Q_3 = \{\langle X, Y\rangle \mid \forall t(\forall A(X)(t.A = 1 \wedge \mu) \Rightarrow \forall A(Y)(t.A = 1 \wedge \mu))\}$
where $\mu = \exists B(B = Gender \wedge t.B = F)$.

Q_3 uses schema information of r whereas Q_2 is independent of any schema.

In practice, the kind of data being analyzed clearly influences the definition of a query. Association rules and implications require binary data while functional dependencies can be defined on arbitrary attribute domains.

Furthermore, external information may also be available and useful to define queries. Moreover, it is not necessary to know neither the schema nor the relation to define a query. The schema and the relation turn out to be necessary when we care about the meaning of the rules.

2.3 $\mathcal{RL}$ semantics

For the sake of completeness, we defined $\mathcal{RL}$ semantics in the classical sense.

Definition 4. Let $R \subseteq \mathcal{U}$, r be a relation over R and $adom(r) \subseteq \mathcal{CST}$ the active domain of r. An *$\mathcal{RL}$-structure* over R is defined by r and an assignment Σ from schema-variables to 2^R. An *$\mathcal{RL}$-structure* over R is denoted by $\langle r, \Sigma\rangle$.

Example 3. Let r_1 be a relation over $R = \{\overline{A}, \overline{B}, \overline{C}\}$ and X, Y two schema-variables.
For instance, consider that $\Sigma_1(X) = \{\overline{A}, \overline{B}\}$ and $\Sigma_1(Y) = \{\overline{C}\}$. $\langle r_1, \Sigma_1\rangle$ is a $\mathcal{RL}$-structure over R.

Definition 5. A *$\mathcal{RL}$-interpretation* over R is defined by a $\mathcal{RL}$-structure $\langle r, \Sigma\rangle$ and an assignment σ to every tuple- and attribute-variable defined as follows:

- Let t be a tuple-variable. $\sigma : t \mapsto$ tuple defined over R
- Let A be an attribute-variable. $\sigma : A \mapsto$ attribute of R

An *$\mathcal{RL}$-interpretation* over R is denoted by $\langle\langle r, \Sigma\rangle, \sigma\rangle$.

Example 4. Continuing the previous example, let s, t be two tuple-variables and r_1 defined as:

r_1	$\overline{A}$	$\overline{B}$	$\overline{C}$
$\overline{t_1}$	1	2	1
$\overline{t_2}$	1	2	3
$\overline{t_3}$	2	2	3
$\overline{t_4}$	3	4	5

Consider the two following assignments σ_1 and σ_2:

- $\sigma_1 : t \mapsto \overline{t_1}, s \mapsto \overline{t_2}$
- $\sigma_2 : t \mapsto \langle 0, 0, 0\rangle, s \mapsto \overline{t_3}$

$\langle\langle r_1, \Sigma_1\rangle, \sigma_1\rangle$ and $\langle\langle r_1, \Sigma_1\rangle, \sigma_2\rangle$ are two $\mathcal{RL}$-interpretation.

Satisfaction of $\mathcal{RL}$-formulas. Given a $\mathcal{RL}$-interpretation, the satisfaction of a $\mathcal{RL}$-formula can now be defined.

Definition 6. Let δ be a $\mathcal{RL}$-formula. The *satisfaction* of δ with respect to a *$\mathcal{RL}$-interpretation* $\langle\langle r, \Sigma\rangle, \sigma\rangle$, denoted by $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \delta$, is defined inductively as follows:

- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models A = B$ iff $\sigma(A) = \sigma(B)$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models t.A = s.B$ iff $\sigma(t)[\sigma(A)] = \sigma(s)[\sigma(B)]$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models t.A = a$ iff $\sigma(t)[\sigma(A)] = a$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models X(A)$ iff $\sigma(A) \in \Sigma(X)$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models X(\overline{A})$ iff $\overline{A} \in \Sigma(X)$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \neg\delta$ iff $\langle\langle r, \Sigma\rangle, \sigma\rangle \not\models \delta$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \delta_1 \wedge \delta_2$ iff $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \delta_1$ and $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \delta_2$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \forall A(\delta)$ iff for every $\overline{A} \in R$, $\langle\langle r, \Sigma\rangle, \sigma_{A \mapsto \overline{A}}\rangle \models \delta$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \forall u(\delta)$ iff for every $\overline{t} \in r$, $\langle\langle r, \Sigma\rangle, \sigma_{u \mapsto \overline{t}}\rangle \models \delta$
- $\langle\langle r, \Sigma\rangle, \sigma\rangle \models t = s$ iff $\sigma(t) = \sigma(s)$

Example 5. Continuing previous examples, let

$$\delta = \forall A(X)(s.A = t.A)$$

We get $\langle\langle r_1, \Sigma_1\rangle, \sigma_1\rangle \models \delta$ since $\overline{t_1}[\overline{A}] = \overline{t_2}[\overline{A}]$ and $\overline{t_1}[\overline{B}] = \overline{t_2}[\overline{B}]$. We also get $\langle\langle r_1, \Sigma_1\rangle, \sigma_2\rangle \not\models \delta$ since $\overline{t_3}[\overline{A}] \neq 0$.

When clear from context, we shall note $\langle r, \sigma\rangle \models \delta$ instead of $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \delta$ whenever δ does not have any schema-variables, i.e. the schema-variable assignment Σ is useless.

2.4 Answer of a $\mathcal{RL}$-query

We can now define the answer of a $\mathcal{RL}$-query.

Definition 7. Let Q_δ be a $\mathcal{RL}$-query and r a relation over R. The answer of Q_δ in r, denoted by $ans(Q_\delta, r)$, is defined as:

$ans(Q_\delta, r) = \{\Sigma(X) \rightarrow \Sigma(Y) \mid \langle r, \Sigma\rangle$ is a $\mathcal{RL}$-structure s.t. $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \delta$ for every $\sigma\}$

In the sequel, $\Sigma(X) \rightarrow \Sigma(Y)$ is referred to as a *rule* and will be also equivalently denoted by the couple $(\Sigma(X), \Sigma(Y))$.

Example 6. Continuing previous examples, we have:

$ans(Q_{f_1}, r_1) =$
{
$\overline{A} \rightarrow \overline{A}, \overline{B} \rightarrow \overline{B}, \overline{C} \rightarrow \overline{C}, \overline{AB} \rightarrow \overline{AB}, \overline{AC} \rightarrow \overline{AC}$,
$\overline{BC} \rightarrow \overline{BC}, \overline{ABC} \rightarrow \overline{ABC}, \overline{ABC} \rightarrow \overline{A}, \overline{ABC} \rightarrow \overline{B}$,
$\overline{ABC} \rightarrow \overline{C}, \overline{ABC} \rightarrow \overline{AB}, \overline{ABC} \rightarrow \overline{AC}, \overline{ABC} \rightarrow \overline{BC}$,
$\overline{AB} \rightarrow \overline{A}, \overline{AB} \rightarrow \overline{B}, \overline{AC} \rightarrow \overline{A}, \overline{AC} \rightarrow \overline{C}, \overline{BC} \rightarrow \overline{B}$,
$\overline{BC} \rightarrow \overline{C}, \overline{A} \rightarrow \overline{B}, \overline{A} \rightarrow \overline{AB}, \overline{AC} \rightarrow \overline{B}, \overline{AC} \rightarrow \overline{BC}$,
$\overline{AC} \rightarrow \overline{AB}, \overline{AC} \rightarrow \overline{ABC}, \overline{C} \rightarrow \overline{B}, \overline{C} \rightarrow \overline{BC}$
}

For convenience, a set $\{\overline{A}\}$ is referred to as $\overline{A}$ and a set $\{\overline{A_1}, \ldots, \overline{A_n}\}$ is referred to as $\overline{A_1, \ldots, A_n}$

Now, we define the notion of satisfaction of a rule in a relation with respect to a $\mathcal{RL}$-query.

Definition 8. Let Q be a $\mathcal{RL}$-query, r a relation over R and $\overline{X} \rightarrow \overline{Y}$ a rule. The *satisfaction* of $\overline{X} \rightarrow \overline{Y}$ in r, denoted by $r \models_Q \overline{X} \rightarrow \overline{Y}$, is defined as:

$$r \models_Q \overline{X} \rightarrow \overline{Y} \text{ iff } \overline{X} \rightarrow \overline{Y} \in ans(Q, r)$$

By extension, let F be a set of rules over R. We note $r \models_Q F$ iff for every $\overline{X} \rightarrow \overline{Y} \in F, r \models_Q \overline{X} \rightarrow \overline{Y}$.

Now, given a $\mathcal{RL}$-query, we can define the notion of *logical implication* of a rule with respect to a set of rules.

Definition 9. Given a $\mathcal{RL}$-query Q, a schema R, a set of rules F over R and a rule $\overline{X} \rightarrow \overline{Y}$, the *logical implication*, denoted by $F \models_Q \overline{X} \rightarrow \overline{Y}$, is defined as:
$F \models_Q \overline{X} \rightarrow \overline{Y}$ iff for all r over R such that $r \models_Q F$, we have: $r \models_Q \overline{X} \rightarrow \overline{Y}$

3. ARMSTRONG-COMPLIANT QUERY LANGUAGE

In order to define a class of queries of interest, we are interested in some axiomatizations to get some reasoning capabilities. Clearly, functional dependencies and implications in formal concept analysis fit into the framework presented so far (cf examples 1, 2). In this paper, we focus on a well-known axiomatization in database and FCA: *Armstrong's axioms* due to its efficiency to check the implication problem [5]. Let us recall the Armstrong's axioms for a set of rules F defined over a set of attributes U:

1. **Reflexivity:** If $\overline{X} \subseteq \overline{Y} \subseteq \mathcal{U}$ then $F \vdash \overline{Y} \rightarrow \overline{X}$
2. **Augmentation:** If $F \vdash \overline{X} \rightarrow \overline{Y}$ and $\overline{W} \subseteq \mathcal{U}$, then $F \vdash \overline{XW} \rightarrow \overline{YW}$
3. **Transitivity:** If $F \vdash \overline{X} \rightarrow \overline{Y}$ and $F \vdash \overline{Y} \rightarrow \overline{Z}$ then $F \vdash \overline{X} \rightarrow \overline{Z}$

The notation $F \vdash \overline{X} \rightarrow \overline{Y}$ means that a derivation of $\overline{X} \rightarrow \overline{Y}$ can be obtained using Armstrong's axiom system from F.

The practical interests of this axiomatization are twofold:

- Firstly, *reasoning* can be performed on rules from the Armstrong's axioms. For instance, the implication problem can be resolved in linear time [5].
- Secondly, instead of listing many redundant rules, we can focus on "small" *covers* of rules [11, 12, 16].

In the setting of this class of queries, *query processing techniques* could be devised, allowing to bridge the gap between data mining and databases. This is out of the scope of this paper though.

The first point is to characterize, for a given $\mathcal{RL}$-query Q, what does that mean that this query "complies" with the Armstrong's axioms ? The first idea is to extend the classical definition as follows: For a given query Q, a schema R and a set of rules F over R, we would like to have $F \vdash \overline{X} \rightarrow \overline{Y}$ iff $F \models_Q \overline{X} \rightarrow \overline{Y}$.

Unfortunately, this definition turns out to be useless as shown in the following example.

Example 7. Let $Q_1 = \{\langle X,\ Y\rangle |\ \ \textbf{true}\ \}$
We have $F \models_{Q_1} \overline{X} \rightarrow \overline{Y}$ for every F and for every $\overline{X} \rightarrow \overline{Y}$. As a consequence, $F \models_{Q_1} \overline{X} \rightarrow \overline{Y}$ does not imply $F \vdash \overline{X} \rightarrow \overline{Y}$.

As a matter of fact, we need to relax somehow the tentative definition of compliance of a query with respect to Armstrong's axioms. To do that, we introduce a set-oriented perspective for Armstrong compliance.

Definition 10. Let $\mathcal{U}$ be a finite set. Let $S \subseteq 2^{\mathcal{U}} \times 2^{\mathcal{U}}$. We say that S is *Armstrong-closed* if and only if it satisfies the following three conditions:

1. **Reflexivity:** For all $\overline{X}, \overline{Y} \subseteq \mathcal{U}$ if $\overline{X} \subseteq \overline{Y}$, then $(\overline{Y}, \overline{X}) \in S$.
2. **Augmentation:** For all $\overline{X}, \overline{Y}, \overline{W} \subseteq \mathcal{U}$, if $(\overline{X}, \overline{Y}) \in S$, then $(\overline{X} \cup \overline{W}, \overline{Y} \cup \overline{W}) \in S$.
3. **Transitivity:** For all $\overline{X}, \overline{Y}, \overline{Z} \subseteq \mathcal{U}$, if $(\overline{X}, \overline{Y}) \in S$ and $(\overline{Y}, \overline{Z}) \in S$, then $(\overline{X}, \overline{Z}) \in S$.

Definition 11. A $\mathcal{RL}$-query Q is *Armstrong-compliant* if and only if for every relation r, $ans(Q, r)$ is Armstrong-closed.

Example 8. Continuing the previous example, Q_1 is Armstrong compliant.

4. NEGATIVE RESULTS

We show here some negative results, mainly the undecidability of Armstrong-compliance of $\mathcal{RL}$-queries and the undecidability of testing the equivalence of two formulas. First, we consider two simple queries and point out their status with respect to Armstrong-compliance.

Lemma 1. Let $Q_1 = \{\langle X,\ Y\rangle |\ \ \textbf{true}\ \}$ and $Q_2 = \{\langle X,\ Y\rangle |\ \ \textbf{false}\ \}$

1. Q_1 is Armstrong-compliant.
2. Q_2 is not Armstrong-compliant.

PROOF. We show the two assertions

1. For every schema R and every relation r over R, all possible rules belong to $ans(Q_1, r)$, i.e. $2^R \times 2^R$. The result follows.
2. For every schema R and every relation r over R, $ans(Q_2, r)$ is empty. Therefore, rules induced by reflexivity do not belong to $ans(Q_2, r)$.

□

Now, it is worth noting that in this paper $\mathcal{RL}$-formulas do not include real attributes (only attribute variables are allowed). As a consequence, $\mathcal{RL}$-formulas cannot simulate the tuple relational calculus, even if this is a simple extension of the $\mathcal{RL}$-language, not described here though. For simplicity of the arguments, we shall assume in the sequel that TRC formulas are allowed in $\mathcal{RL}$-formulas.

Theorem 1 *Armstrong-compliance of $\mathcal{RL}$-queries is undecidable.*

PROOF. Consider $\mathcal{RL}$-query of the form:

$$Q_\varphi = \{(X,Y) | \varphi \wedge \forall A(X)\ \textbf{true} \Rightarrow \forall A(Y)\ \textbf{true}\}$$

where φ is a closed formula in tuple relational calculus. Assume relations r over a finite set of attributes $\overline{U}$. We can notice that:

1. if φ is logically valid (i.e. true under every possible interpretation), then for every relation r, $ans(Q_\delta, r) = ans(Q_1, r)$. By lemma 1, Q_δ is Armstrong-compliant.
2. if φ is not logically valid, then there exists a relation r such that r is not satisfied by φ. Then, we have $ans(Q_\delta, r) = ans(Q_2, r)$. By lemma 1, Q_δ is not Armstrong-compliant.

Consequently, $Q_\varphi(r)$ is Armstrong-compliant if and only if φ is logically valid. Yet, it is undecidable to know whether or not a given closed formula is logically valid ([7], theorem 6.3.1 in [1]). □

Next, we define what does equivalence mean in our context. Then, we show a negative computational property, i.e. checking predicate equivalence turns out to be undecidable.

We shall define the notion of equivalence between $\mathcal{RL}$-formulas as follows:

Definition 12. Two $\mathcal{RL}$-formulas δ_1 and δ_2 are said to be *equivalent*, denoted $\delta_1 \equiv \delta_2$, iff for every $\mathcal{RL}$-interpretation $\langle\langle r, \Sigma\rangle, \sigma\rangle$ such that $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \delta_1$, then $\langle\langle r, \Sigma\rangle, \sigma\rangle \models \delta_2$ and vice versa.

Theorem 2 *Let δ_1 and δ_2 be two $\mathcal{RL}$-formulas. Checking equivalence between δ_1 and δ_2 is undecidable.*

The proof is omitted.

5. TOWARDS AN ARMSTRONG-COMPLIANT $\mathcal{RL}$-LANGUAGE

From the negative results shown so far, we are going to restrict the "shape" of the query language defined previously (cf. definition 3).

Impact of the $\mathcal{RL}$-formula $\psi(X, Y, \vec{t})$. The previous undecidability result (cf theorem 1) points out that $\psi(X, Y, \vec{t})$ should be removed from the query language. First, we show that this constraint is indeed useless in our setting.

Lemma 2. Let Q be a $\mathcal{RL}$-query. If Q is Armstrong-compliant then ψ is logically valid.

PROOF. Now suppose that Q is Armstrong-compliant and $\psi(X, Y, \vec{t})$ is not logically valid.
Therefore, there exists at least one $\mathcal{RL}$-interpretation $\langle\langle r, \Sigma\rangle, \sigma\rangle$ such that $\langle\langle r, \Sigma\rangle, \sigma\rangle \not\models \psi(X, Y, \vec{t})$. Therefore, $ans(Q, r)$ is empty and by the Lemma 1, Q is not Armstrong-compliant. Contradiction. □

We deduce from this result that $\psi(X, Y, \vec{t})$ is useless to get Armstrong-compliant queries since ψ should be necessary logically valid.

Unfortunately, this is not sufficient to be sure to define only Armstrong-compliant queries. In the sequel, we exhibit counter-examples from which a slight restriction of $\mathcal{RL}$-queries will be proposed.

Impact of existential quantifiers. Let us consider an example of queries in which existential quantifiers are used in δ_1 and δ_2.

Example 9.

$Q'_\delta = \{\langle X, Y\rangle | \forall t, s(\exists A(X)(t.A = s.A)) \Rightarrow (\exists B(Y)(t.B = s.B))\}$

Let r_0 be the following relation:

r_0	$\overline{A}$	$\overline{B}$
$\overline{t_1}$	0	0
$\overline{t_2}$	0	1

Let us consider the following assignments over r_0 : $\Sigma_1(X) = \{\overline{A, B}\}, \Sigma_1(Y) = \{\overline{B}\}$ and $\sigma_1(t) = \overline{t_1}, \sigma_1(s) = \overline{t_2}, \sigma_1(A) = \overline{A}, \sigma_1(B) = \overline{B}$.

Then, $\overline{A, B} \rightarrow \overline{B} \notin ans(Q'_\delta, r_0)$ and the reflexivity is lost.

The previous example suggests to allow only universal quantifiers in δ_1 and δ_2 of $\mathcal{RL}$-queries.

Impact of different formulas in the left- and right-hand sides of a rule. The next example stresses the need to have equivalent predicates δ_1 and δ_2.

Example 10. Let us consider the following query.

$Q"_\delta = \{\langle X, Y\rangle \mid \forall t(\forall A(X)(t.A = 1) \Rightarrow \forall B(Y)(t.B = 0))\}$

Let r be the following relation:

r	$\overline{A}$	$\overline{B}$	$\overline{C}$
$\overline{t_1}$	1	0	1
$\overline{t_2}$	0	1	0

We have $r \models_{Q"_\delta} \overline{A} \rightarrow \overline{B}$, $r \models_{Q"_\delta} \overline{B} \rightarrow \overline{C}$ but $r \not\models_{Q"_\delta} \overline{A} \rightarrow \overline{C}$, cf tuple $\overline{t_1}$ (lost of the transitivity).

Nevertheless, checking the equivalence of δ_1 and δ_2 has been shown to be undecidable (cf theorem 2).

5.1 The $\mathcal{RLR}$ language

In the setting of the query language defined in definition 3, we made syntactic restrictions to obtain the $\mathcal{RLR}$-query language. The main result of the paper will be given on that class of queries.

Definition 13. The $\mathcal{RLR}$-query language is defined as the set of $\mathcal{RLR}$-queries Q_δ expressed as follows:

$$Q_\delta = \{\langle X, Y\rangle \mid \underbrace{\forall \vec{t}(\forall A(X)\delta_1(A, \vec{t}) \Rightarrow \forall A(Y)\delta_1(A, \vec{t}))}_{\delta(X,Y)}\}$$

where:

- X, Y are free schema-variables on $\delta(X, Y)$.
- $\forall \vec{t}$ is a vector of universally quantified tuple-variables. Let say $t = \langle t_1, \ldots, t_n\rangle$
- $\delta_1(A, \vec{t})$ is a $\mathcal{RL}$-formula with $A, t_1, \ldots, t_n$ free variables and without other free variables.

The main result of the paper can be given on the class of $\mathcal{RLR}$-queries, i.e. they enjoy Armstrong-compliance.

This result is stated as follows:

Theorem 3 *If Q is a $\mathcal{RLR}$-query, then Q is Armstrong-compliant.*

PROOF. (reflexivity) Let us assume $\overline{Y} \subseteq \overline{X}$ and let r_0 be a relation over R. We have to show that $(\overline{X}, \overline{Y}) \in ans(Q, r_0)$ i.e. $r_0 \models_Q \overline{X} \rightarrow \overline{Y}$, or $\overline{X} \rightarrow \overline{Y} \in ans(Q, r_0)$.
Assume $\langle r_0, \Sigma \rangle$ is a $\mathcal{RLR}$-structure with $\Sigma(X) = \overline{X}$ and $\Sigma(Y) = \overline{Y}$ such that $\langle \langle r_0, \Sigma \rangle, \sigma \rangle \models \forall \vec{t}(\forall A(X)\delta_1(A, \vec{t}))$ for every σ.
By assumption, we know that $\overline{Y} \subseteq \overline{X}$. Then we have $\langle \langle r_0, \Sigma \rangle, \sigma \rangle \models \forall \vec{t}(\forall A(Y)\delta_1(A, \vec{t}))$ for every σ.
So $\overline{X} \rightarrow \overline{Y} \in ans(Q, r_0)$ and $(\overline{X}, \overline{Y}) \in ans(Q, r_0)$.

(augmentation) Let r_0 be a relation over R such that $(\overline{X}, \overline{Y}) \in ans(Q, r_0)$ and $\overline{W} \subseteq \mathcal{U}$. We have to show that $(\overline{XW}, \overline{YW}) \in ans(Q, r_0)$ i.e. $r_0 \models_Q \overline{XW} \rightarrow \overline{YW}$, or $\overline{XW} \rightarrow \overline{YW} \in ans(Q, r_0)$.
Assume $\langle r_0, \Sigma \rangle$ is a $\mathcal{RLR}$-structure with $\Sigma(X) = \overline{X}$, $\Sigma(Y) = \overline{Y}$ and $\Sigma(W) = \overline{W}$ such that $\langle \langle r_0, \Sigma \rangle, \sigma \rangle \models \forall \vec{t}(\forall A(XW)\delta_1(A, \vec{t}))$ for every σ.
By assumption, we know that $(\overline{X}, \overline{Y}) \in ans(Q, r_0)$ i.e. $r_0 \models_Q \overline{X} \rightarrow \overline{Y}$. Then we have $\langle \langle r_0, \Sigma \rangle, \sigma \rangle \models \forall \vec{t}(\forall A(YW)\delta_1(A, \vec{t}))$ for every σ.
So $\overline{XW} \rightarrow \overline{YW} \in ans(Q, r_0)$ and $(\overline{XW}, \overline{YW}) \in ans(Q, r_0)$.

(transitivity) Let r_0 be a relation over R such that $(\overline{X}, \overline{Y}) \in ans(Q, r_0)$ and $(\overline{Y}, \overline{Z}) \in ans(Q, r_0)$. We have to show that $(\overline{X}, \overline{Z}) \in ans(Q, r_0)$ i.e. $r_0 \models_Q \overline{X} \rightarrow \overline{Z}$, or $\overline{X} \rightarrow \overline{Z} \in ans(Q, r_0)$.
Assume $\langle r_0, \Sigma \rangle$ is a $\mathcal{RLR}$-structure with $\Sigma(X) = \overline{X}$, $\Sigma(Y) = \overline{Y}$ and $\Sigma(Z) = \overline{Z}$ such that $\langle \langle r_0, \Sigma \rangle, \sigma \rangle \models \forall \vec{t}(\forall A(X)\delta_1(A, \vec{t}))$ for every σ.
By assumption, we know that $(\overline{X}, \overline{Y}) \in ans(Q, r_0)$ i.e. $r_0 \models_Q \overline{X} \rightarrow \overline{Y}$. Then we have $\langle \langle r_0, \Sigma \rangle, \sigma \rangle \models \forall \vec{t}(\forall A(Y)\delta_1(A, \vec{t}))$ for every σ.
Moreover, $(\overline{Y}, \overline{Z}) \in ans(Q, r_0)$ i.e. $r_0 \models_Q \overline{Y} \rightarrow \overline{Z}$. Then we have $\langle \langle r_0, \Sigma \rangle, \sigma \rangle \models \forall \vec{t}(\forall A(Z)\delta_1(A, \vec{t}))$ for every σ.
So $\overline{X} \rightarrow \overline{Z} \in ans(Q, r_0)$ and $(\overline{X}, \overline{Z}) \in ans(Q, r_0)$.

□

5.2 Computational property

We proved that $\mathcal{RLR}$-queries are Armstrong-compliant. Given a $\mathcal{RLR}$-query Q and a database r, how can we compute $ans(Q, r)$? We just sketch one easy result, this issue being out of the scope of this paper.

It is worth noting that this classical database problem can be seen de facto as both a pattern mining problem in data mining and an enumeration problem in combinatorics. We give a result related to levelwise strategies in data mining [3, 17].

Property 1 *$\mathcal{RLR}$-queries are amenable to levelwise strategies.*

PROOF. Let Q be a $\mathcal{RLR}$-query. We have to show that for every r over R, if $\overline{X} \rightarrow \overline{Y} \in ans(Q, r)$ then for all $\overline{Z} \supset \overline{X}, \overline{Z} \rightarrow \overline{Y} \in ans(Q, r)$. The result follows since Q is Armstrong-compliant. □

5.3 Example of $\mathcal{RLR}$-query

We give below an example of $\mathcal{RLR}$-query that might be useful in practice.

Example 11. Let $Q_{efd} = \{\langle X, Y \rangle | \forall t_1, t_2, t_3(\delta(X, t_1, t_2, t_3)) \Rightarrow \delta(Y, t_1, t_2, t_3)\}$

where

$\delta(X, t_1, t_2, t_3) = \forall A(X)(t_1.A = t_2.A \wedge t_3.A = t_2.A \wedge \neg(t_1 = t_2) \wedge \neg(t_2 = t_3) \wedge \neg(t_1 = t_3))$

Q_{efd} extends the definition of FD with three tuples instead of two. Let r_0 be the following relation:

r_0	$\overline{A}$	$\overline{B}$	$\overline{C}$	$\overline{D}$
	0	1	1	1
	0	1	2	1
	0	1	3	2
	1	2	4	2
	1	1	5	3

We can see that Q_{efd} is a $\mathcal{RLR}$-query and consequently is Armstrong-compliant.

Let $ans^{min}(Q_{efd}, r_0)$ (resp. $ans^{min}(Q_{fd}, r_0)$) be a smallest equivalent subset of $ans(Q_{efd}, r_0)$ (resp. $ans(Q_{fd}, r_0)$).

Here we have:
$ans^{min}(Q_{efd}, r_0) = \{\overline{A} \rightarrow \overline{B}, \overline{B} \rightarrow \overline{A}, \overline{C} \rightarrow \overline{ABD}, \overline{D} \rightarrow \overline{ABC}\}$

$ans^{min}(Q_{fd}, r_0) = \{\overline{C} \rightarrow \overline{ABD}, \overline{AD} \rightarrow \overline{B}, \overline{BD} \rightarrow \overline{A}\}$.

6. CONCLUSION

In this article, we have proposed an extension of the relational calculus and we have introduced $\mathcal{RL}$-language, a simple logical query language that allows to express different "kinds of rule". We have shown some undecidability results related to $\mathcal{RL}$ -queries and Armstrong-compliance.

We have bounced on this result and have defined the $\mathcal{RLR}$ -language as a restriction of the $\mathcal{RL}$-language. We have pointed out that queries of this sublanguage "enjoy" Armstrong's axioms. From the last result, we have easily deduced $\mathcal{RLR}$-queries are amenable to a levelwise strategy, i.e. the Apriori trick can be used to compute the result of $\mathcal{RLR}$-queries.

Many open questions remain to be addressed.

First, we focus on this paper on a very simple logical query language, which may appear to be a bit restrictive to be useful in practice. Nevertheless, it is worth noting that more expressivity is possible for instance by defining weaker form of equality in atomic formulas such as $|A - B| \leq \epsilon$ for handling numerical attributes.

Then, our language offers new opportunities to apply database query processing techniques in a data mining setting. Arm-

strong's axioms could be used to enumerate efficiently the results of $\mathcal{RLR}$-queries.

One may also question about a larger class of Armstrong-compliant queries. Clearly, some queries not in $\mathcal{RLR}$-language still satisfy Armstrong's axioms. A toy example is $Q = \{\langle X, Y\rangle \mid \forall t(\forall A(X)(t.A = 1) \Rightarrow \forall B(Y)(\neg(\neg(t.B = 1))))\}$.

Finally, other axiomatizations than Armstrong's axioms could be chosen as target of some new query languages.

7. ACKNOWLEDGMENTS

We would like to thank the anonymous referees for their helpful suggestions and comments.

This work was partially supported by the ANR (french National Research Agency) project DAG (ANR-09-DEFIS, 2009-2012).

8. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[2] M. Agier, J.-M. Petit, and E. Suzuki. Unifying framework for rule semantics: Application to gene expression data. *Fundam. Inform.*, 78(4):543–559, 2007.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD International Conference on Management of Data, Washington D.C.*, pages 207–216. ACM Press, 1993.

[4] W. W. Armstrong. Dependency structures of data base relationships. In *Proc. of IFIP Congress*, pages 580–583, 1974.

[5] C. Beeri and P. Berstein. Computational problems related to the design of normal form relation schemes. *ACM Trans. Database Syst.*, 4(1):30–59, 1979.

[6] T. Calders and J. Wijsen. On monotone data mining languages. In G. Ghelli and G. Grahne, editors, *DBPL*, volume 2397 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2001.

[7] A. Church. A note on the entscheidungsproblem. *J. Symb. Log.*, 1(1):40–41, 1936.

[8] L. Fang and K. LeFevre. Splash: ad-hoc querying of data and statistical models. In *EDBT*, pages 275–286, 2010.

[9] B. Ganter and R. Wille. *Formal Concept Analysis*. SPRINGER, 1999.

[10] F. Giannotti, G. Manco, and F. Turini. Towards a logic query language for data mining. In *Database Support for Data Mining Applications, LNCS 2682*, pages 76–94, 2004.

[11] G. Gottlob and L. Libkin. Investigations on Armstrong relations, dependency inference, and excluded functional dependencies. *Acta Cybernetica*, 9(4):385–402, 1990.

[12] J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Math. Sci. Humaines*, 24(95):5–18, 1986.

[13] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Commun. ACM*, 39(11):58–64, 1996.

[14] M. Lacroix and A. Pirotte. Domain-oriented relational languages. In *Proceedings of the third international conference on Very large data bases*, volume 3, pages 370–378. VLDB Endowment, 1977.

[15] H.-C. Liu, A. Ghose, and J. Zeleznikow. Towards an algebraic framework for querying inductive databases. In *DASFAA (2)*, pages 306–312, 2010.

[16] D. Maier. Minimum covers in the relational database model. *J. ACM*, 27(4):664–674, 1980.

[17] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258, 1997.

[18] R. Meo, G. Psaila, and S. Ceri. An extension to sql for mining association rules. *Data Min. Knowl. Discov.*, 2(2):195–224, 1998.

[19] A. Netz, S. Chaudhuri, J. Bernhardt, and U. M. Fayyad. Integration of data mining with database technology. In *VLDB*, pages 719–722, 2000.

[20] L. D. Raedt. An inductive logic programming query language for database mining. In *AISC*, pages 1–13, 1998.

[21] L. D. Raedt, T. Guns, and S. Nijssen. Constraint programming for itemset mining. In *KDD*, pages 204–212, 2008.